LabVIEW™ Microprocessor SDK Release Notes

Version 8.6

Contents

NATIONAL INSTRUMENTS™

Porting LabVIEW to a New Embedded Target

Use the LabVIEW Microprocessor SDK to port LabVIEW to any 32-bit microprocessor. By using a single development tool from concept to finished product, you can ease the development process and increase end quality while reducing time to market.

The porting process includes several steps; some steps are required and some steps are optional depending on your target and the features you want to implement and support.

The main steps to porting LabVIEW include the following:

1. Obtaining the necessary toolchain and board support package (BSP) for your target.
2. Compiling, downloading, running, and debugging a “hello world” application using the toolchain and BSP. This step verifies that the necessary toolchain is installed on the host computer and the board support package is installed and configured correctly.
3. Porting the LabVIEW Run-Time Library to the target operating system.
4. Creating and/or modifying the plug-in VIs for basic user actions and target-specific dialog boxes.
5. Adding the target to the LabVIEW development environment.

The actual amount of implementing versus reusing of example targets depends on how closely your target, operating system, and toolchain match one of the example targets, operating systems, and toolchains. While the example targets include common processor architectures, operating systems, and toolchains, the examples targets cannot cover everything.

The Microprocessor SDK includes the LabVIEW C Code Generator, which generates C code based on the block diagram when you build an embedded VI into an embedded application. Next, the C code is passed with any external C code and the LabVIEW Run-Time Library through your third-party cross-compiler to create an executable file. This executable file is saved on the host computer.

When you download, or deploy, an embedded application, your toolchain downloads the application, usually over serial, TCP, or JTAG. If you run the embedded application, the go command is sent for that application. A basic embedded application runs headless, which means it runs without a user interface, keyboard, mouse, and so on. If your target has an LCD and you implement user interface support, your embedded application might have a user interface. When you debug an embedded application, you create an interactive debug connection back to the host PC, usually over serial, TCP, or JTAG.

System Requirements and Target Recommendations

Host Computer Requirements

The Microprocessor SDK has the following requirements:

- A computer with Windows Vista/XP/2000
- LabVIEW 8.6 Full or Professional Edition
- NI-VISA Run-Time Engine version 3.1 or later (available for download at `ni.com`)

Refer to the *LabVIEW Release Notes*, available by selecting **Start»All Programs»National Instruments»LabVIEW 8.6» LabVIEW Manuals** and opening `LV_Release_Notes.pdf`, for standard LabVIEW development system requirements.

Target Recommendations

National Instruments recommends that your target has the following:

- 32-bit processor architecture.
- 256 KB of application memory (in addition to the embedded operating system memory requirements).
- An embedded operating system so you can take advantage of the parallelism and multithreading in LabVIEW.
- A hardware floating-point unit or floating-point emulation library to perform the math calculations in the LabVIEW Analysis Library, which is floating-point. You can improve performance by performing the math calculations in hardware instead of using software emulation.

System Requirements for Example Targets

The Microprocessor SDK example targets have the following requirements:

Target Name	Hardware Requirements	Software Requirements
Code Generation Only	None	None
Axiom CMD565, eCos RAM Image	Axiom CMD-565 Development Board	Cygwin 1.5.*x*, eCos 2.0 PowerPC toolchain
Axiom CMD565, eCos ROM Image	Axiom CMD-565 Development Board	Cygwin 1.5.*x*, eCos 2.0 PowerPC toolchain
Axiom CMD565, VxWorks Module	Axiom CMD-565 Development Board	Wind River Tornado 2.2.1, VxWorks 5.5.1 BSP for CMD565

Target Name	Hardware Requirements	Software Requirements
Axiom CMD565, VxWorks RAM Image	Axiom CMD-565 Development Board, iSYSTEM iC3000ActiveEmulator	Wind River Tornado 2.2.1, VxWorks 5.5.1 BSP for CMD565, iSYSTEM winIDEA
Axiom CMD565, VxWorks ROM Image	Axiom CMD-565 Development Board	Wind River Tornado 2.2.1, VxWorks 5.5.1 BSP for CMD565
VxWorks Simulation	None	Wind River Tornado 2.2.1, VxWorks 5.5.1, (Optional) ULIP Ethernet driver
Freescale ColdFire M5329EVB, uClinux	ColdFire M5329EVB Development Kit	CodeSourcery ColdFire uClinux toolchain release 4.1-11 m68k, CF Flasher 3.1, P&E Device Drivers
PHYTEC LPC229 *x*, eCos	phyCORE-ARM7/LPC229*x* Rapid Development Kit, GPIO Expansion Board	Cygwin 1.5.*x*, eCos 2.0 ARM toolchain
Spectrum Digital DSK6713, DSP/BIOS	DSP Starter Kit (DSK) for the TMS320C6713	TI Code Composer Studio 3.1
Unix Console	None	Cygwin 1.5.*x* with gcc package
Unix UI	None	Cygwin 1.5.*x* with gcc package
Unicon UCN2410-CWIFI, Linux	Mobile Development Kit MKit UCN2410-CWIFI	Cygwin 1.5.*x*, FLTK toolchain
Windows Console Application	None	Visual Studio 7.0

Contact the respective vendors for more information about their hardware and software products.

Refer to `ecos.sourceware.org/getstart.html` for information about downloading and installing eCos.

Refer to `gcc.gnu.org` for information about downloading and installing gcc.

The VxWorks for LabVIEW Embedded Development Module Evaluation Kit includes the Tornado 2.2.1 integrated development environment and evaluation run-times for VxWorks 5.5.1 for the purpose of demonstrating the features, performance, and capabilities of these Wind River products in association with the LabVIEW Microprocessor SDK. Refer to `windriver.com/alliances/eval-cd`, click **National Instruments**

Evaluation CD Program, and follow the instructions to receive the VxWorks for LabVIEW Embedded Development Module Evaluation Kit. Refer to windriver.com for more information about Wind River's Device Software Optimization products, including VxWorks real-time operating systems and Tornado, an integrated development environment.

Installation

Complete the following steps to install the Microprocessor SDK.

1. Log on as an administrator or as a user with administrator privileges.
2. Install LabVIEW 8.6, if not already installed.
3. Install the Microprocessor SDK.
4. Activate the Microprocessor SDK.

   You must activate the Microprocessor SDK to access the example targets. You have the option of activating the Microprocessor SDK at the end of installation. You also can use the NI License Manager, available by selecting **Start»All Programs»National Instruments» NI License Manager**, to activate National Instruments products. Refer to the *National Instruments License Manager Help*, available by selecting **Help»Contents** in the National Instruments License Manager, for more information about activating NI products.
5. Restart the computer when the installer completes.

Tip If you have beta versions, previous versions, and/or multiple versions of the Embedded Development Module and/or Microprocessor SDK and the activation is not successful at the end of installation, use the NI License Manager to activate the software.

Selecting an Appropriate Example Target

Selecting an appropriate example target is a good place to start when you port LabVIEW to a new embedded target. Use the target that is closest to your target and toolchain. For example, if you are using a GNU C/C++-based (gcc) toolchain, consider using an eCos target. If you are using a VxWorks-based toolchain with different hardware, you might want to use a VxWorks subtarget. Subtargets are targets that reuse existing functionality from another target that uses the same operating system.

The Microprocessor SDK also includes a blank template target. This target is not intended to be an implementation example, but the target can serve as a good starting point when none of the example targets are appropriate. National Instruments recommends you use the blank target when porting LabVIEW to a new operating system.

None of the example targets are meant to be fully featured, ready to use targets. The different example targets have different implementations. When you are implementing a feature for a new embedded target, look for an existing implementation in an existing example target that is similar to your target. Depending on the feature, you also might want to look for an existing implementation that uses the same operating system as your target.

The following table lists which example targets contain example implementations of different features. Use this table to find an example of a feature you are implementing for your target.

Target Name	Instrumented Debugging	On Chip Debugging	Pre-Built Run-Time Library	Static Memory Model	Memory Mapping	Elemental I/O	IDE Integration	UI
Code Generation Only	No	No	No	Yes	No	No	No	No
Axiom CMD565, eCos RAM Image	Serial	No	No	Yes	No	No	No	No
Axiom CMD565, eCos ROM Image	No	No	No	Yes	No	No	No	No
Axiom CMD565, VxWorks Module	Serial	Wind River WTX	No	Yes	No	No	No	No
Axiom CMD565, VxWorks RAM Image	No	iSYSTEM iC3000	No	Yes	No	No	No	No
Axiom CMD565, VxWorks ROM Image	No	No	No	Yes	Yes	No	No	No
VxWorks Simulation	TCP	No	No	Yes	No	No	No	No
Freescale ColdFire M5329EVB, uClinux	TCP	No	Yes	No	No	No	No	No
PHYTEC LPC229*x*, eCos	Serial	No	Yes	No	Yes	Yes	No	No
Spectrum Digital DSK6713, DSP/BIOS	RTDX	No	Yes	No	No	No	No	No

Target Name	Instrumented Debugging	On Chip Debugging	Pre-Built Run-Time Library	Static Memory Model	Memory Mapping	Elemental I/O	IDE Integration	UI
Unicon UCN2410-CW IFI, Linux	TCP	No	Yes	No	No	No	No	Yes
Unix Console	TCP	Eclipse	Yes	No	No	Simulated	Eclipse	No
Unix UI	TCP	No	Yes	No	No	No	No	Yes
Windows Console Application	TCP	No	Yes	Yes	No	Simulated	No	No

Upgrading from Version 2.5

Changes in the LabVIEW Development System

Refer to the *LabVIEW Release Notes* and *LabVIEW Upgrade Notes*, available by selecting **Start»All Programs»National Instruments» LabVIEW 8.6»LabVIEW Manuals**, for information about new features and changes in the LabVIEW development system.

Upgrading Target Support

Because each target implementation is different, upgrading target support is different for every target. In general, you must complete the following steps to upgrade your target support.

1. Create a folder in `LabVIEW 8.6\Targets`. National Instruments recommends using some version of your company name.
2. Create an **Embedded** folder under the `LabVIEW 8.6\Targets\`*`company name`* directory.

Note LabVIEW does not recognize any targets outside of this directory.

3. Copy your target directory from LabVIEW 8.5, which is located in `LabVIEW 8.5\Targets\`*`company name`*`\Embedded`, to the new LabVIEW 8.6 directory you created in steps 1 and 2.
4. Select **Tools»Microprocessor SDK»Target Editor** in LabVIEW 8.6 to launch the Target Editor and modify the `TgtSupp.xml` file. LabVIEW uses the `TgtSupp.xml` file to incorporate your target into LabVIEW.
5. Open all of your existing plug-in VIs in LabVIEW 8.6 and verify none of the VIs are broken.
6. Move any target-specific VIs and palettes from the LabVIEW 8.5 directory to the LabVIEW 8.6 directory.

Instrumented Debugging Diagnostic Filename Change

You can configure LabVIEW to create an embedded debugging diagnostic file by adding a token to the `LabVIEW.ini` file, which is located in the `labview` directory. The token has changed from `LogPDAMessages=True` to `LogCGenMessages=True`. The resulting filename has changed from `pdamsglog.txt` to `cgenlog.txt`. This text file contains various debugging diagnostic and error messages that occur as a result of instrumented debugging connections. Use this file to help you implement and troubleshoot your instrumented debugging implementation.

What's New

New Unix UI and Unicon Example Targets

The Microprocessor SDK includes two new example targets:

- Unix UI
- Unicon UCN2410-CWIFI

Both example targets include an example implementation of the new user interface support. Refer to the *Microprocessor SDK Porting Guide*, available by selecting **Start»All Programs»National Instruments»LabVIEW»LabVIEW Manuals** and opening `MicroprocessorSDK_Porting_Guide.pdf`, for target set up instructions.

Implementing User Interface Support

If your target has an LCD, you can implement user interface support. The Unix UI and Unicon example targets implement user interface support. The Unix UI example target is similar to the Unix Console example target.

The example user interface support uses FLTK, which is an open source C++ graphical user interface toolkit. You must develop your own video driver if you want to port user interface support and use something other than FLTK.

To implement user interface support, you must do the following:

1. Port the following three open source libraries to your target:
   - **Nano-X (version 0.91)**—low-level screen drawing functions

     You might have to develop a screen driver to work with your target. Also, Nano-X assumes you have a mouse, so if your target uses a touchscreen for input, you also have to develop a touchscreen driver that looks like a mouse to Nano-X. Refer to the Nano-X Web site at `www.microwindows.org` for more information about and documentation for Nano-X.

- **nxLib (version 0.46)**—interface between FLTK and Nano-X
- **FLTK (version 2.0.x-r5940)**—C++ graphics library

  FLTK assumes you have certain header files with certain functions. If these functions do not exist for your target, you must find a workaround. Refer to the FLTK Web site at www.fltk.org for more information about FLTK.

The source code for these libraries are located in the following directory in the `labview\CCodeGen\external` directory.

2. After building the libraries, run and debug the libraries using your toolchain.
3. Build the LabVIEW C Code Run-Time Library and the fltkapi library, which is the interface between FLTK and the LabVIEW C Code Run-Time Library. The source code for the fltkapi library also is located in the `labview\CCodeGen\external` directory.

   You must define the following flags when you build the LabVIEW C Code Run-Time Library:

   - `GUISupport`—Enables user interface support.
   - `FltkGUI`—Enables support for the FLTK widget library.
   - `Supports3DControls`—Enables support for 3D controls.
   - `SupportsColorBG`—Enables support for background color.
   - `FP_SIZE_HORIZ`—Specifies the horizontal size in pixels of the target user interface.
   - `FP_SIZE_VERT`—Specifies the vertical size in pixels of the target user interface.
4. Implement user interface settings in your **Build Specification Properties** dialog box.

Fixed-Point Support

The fixed-point data type has limited support.

Note Overflow mode is supported, but overflow status is not supported.

Supported Numeric Functions

The following **Numeric** functions support the fixed-point data type:

- Absolute Value
- Add
- Decrement
- Increment
- Multiply

- Negate
- Round To Nearest
- Round Toward +Infinity
- Round Toward –Infinity
- Scale By Power Of 2 Function
- Sign
- Subtract
- Square

Comparison Functions

The following **Comparison** functions support the fixed-point data type:

- Equal?
- Equal To 0?
- Greater Or Equal?
- Greater Or Equal To 0?
- Greater?
- Greater Than 0?
- Less Or Equal?
- Less Or Equal To 0?
- Less?
- Less Than 0?
- Not Equal?
- Not Equal To 0?

Conversion Functions

The following **Conversion** functions support the fixed-point data type:

- Boolean Array To Number
- Number To Boolean Array
- To Byte Integer
- To Double Precision Float
- To Extended Precision Float
- To Fixed-Point
- To Long Integer
- To Quad Integer
- To Single Precision Float

- To Unsigned Byte Integer
- To Unsigned Long Integer
- To Unsigned Quad Integer
- To Unsigned Word Integer
- To Word Integer

Data Manipulation Functions

The following **Data Manipulation** functions support the fixed-point data type:

- Flatten To String
- Logical Shift
- Rotate Left With Carry
- Rotate Right With Carry
- Type Cast
- Unflatten From String

String/Number Conversion Functions

The following **String/Number Conversion** functions support the fixed-point data type:

- Decimal String To Number
- Fract/Exp String To Number
- Hexadecimal String To Number
- Number To Decimal String
- Number To Engineering String
- Number To Exponential String
- Number To Fractional String
- Number To Hexadecimal String
- Number To Octal String
- Octal String To Number

New VI-Specific Code Generation Options

You can now optimize subVI calls and inline subVIs into callers, which can eliminate overhead and increase code optimization.

- **Optimize subVI calls**—Generates C code for subVI calls with as little default data initialization as possible. You cannot debug a VI with optimized subVI calls.

- **Allow inlining**—Allows inlining of subVIs into callers. Inlining subVIs is most useful for small subVIs, VIs with many calls in a loop, or subVIs with only one call site. The default is **True.** This option only allows inlining. To actually inline a subVI, you must select **True** from the **Inline subVI** pull-down menu on the **Source File Settings** page in the **Build Specification** dialog box.

For your target to support VI-specific code generation options, you must call `LEP_Utilities_GetProjectVISettings.vi`, located in the `labview\vi.lib\Platform\EmbProject\utilities` directory, from `LEP_x_CGen.vi`, which is located in your target directory. `LEP_Utilities_GetProjectVISettings.vi` builds a list of all VIs and related code generation settings in a project, which becomes an attribute to the variant that determines how the LabVIEW C Code Generator generates the C code from the block diagram.

To set code generation options for a VI, select **File»VI Properties** from the front panel window or block diagram window to open the **VI Properties** dialog box. Select **C Code Generation Option**s from the **Category** pull-down menu. You can select **From project**, **True**, or **False**. Select **True** to enable the option and **False** to disable it. The default is **From project**, so you only need to set the options in VI Properties if you want to override the code generation settings in the project.

New Project-Level Code Generation Options

You now can allocate constants for arrays, clusters, strings, variants, and waveforms with build options on the **Application Information** page in the **Build Specification Properties** dialog box.

Using stack variables overrides constant allocation. If you place a checkmark in the **Use stack variables** checkbox, **Allocate constants** is always **First Use** and **Deallocate constants** is always **Out of Scope**. **First Use** and **Out of Scope** was the default behavior in previous versions.

- **Allocate constants**—Specifies when LabVIEW allocates memory for constants.
  - **First Use**—Allocates memory the first time you use constants on the block diagram.
  - **Containing Loop**—Allocates memory outside of the loop that contains constants.
  - **VI Initialization**—Allocates memory when the VI that contains the constants is called.
  - **Application Initialization**—Allocates memory when the built application begins running on the target.

- **Deallocate constants**—Specifies when LabVIEW frees memory resources for constants.
  - **Out of Scope**—Frees memory resources when the constants are no longer used.
  - **VI End**—Frees memory resources when a VI containing constants finishes executing.
  - **Application End**—Frees memory resources when the built application finishes executing on the target.

Updated Embedded Automatic Test Framework

Two new tokens are available in the `LEP_AutoTest.ini` file, which is located in the `labview\autotest` directory.

- `ReportFolder`—Specifies the path to a summary report that lists which targets have been tested, the number of failed tests per target, and the memory leaks per target.
- `ReportFileName`—Specifies the filename for the summary report.

C Generator-Specific VI Analyzer Tests

The Microprocessor SDK includes VI Analyzer tests for C code generation options and performance.

Select **Tools»VI Analyzer»Analyze VIs** to open the VI Analyzer. Refer to the test descriptions in the VI Analyzer for more information about the tests.

Note You have access only to the C Generator-specific tests unless you install and activate the LabVIEW VI Analyzer Toolkit.

New VIs and Functions

Refer to the *LabVIEW Help* for more information about the new and newly supported VIs and functions.

New Memory Access VIs

The Microprocessor SDK now includes a **Memory Access** palette, which includes the following VIs:

- CCG Peek 8
- CCG Peek 16
- CCG Peek 32
- CCG Poke 8
- CCG Poke 16
- CCG Poke 32

Use these VIs to read and write values to specific memory address locations.

New Console Output VI

The Microprocessor SDK now includes a CCG Console Output VI for `printf` functionality. Use this VI to print text to the standard output stream, `stdout`, of the operating system on the target.

In Place Element Structure Support

The Microprocessor SDK now supports the In Place Element structure, which controls how the LabVIEW compiler performs certain operations and, in some cases, increases memory and VI efficiency.

Synchronization Functions Support

The Microprocessor SDK supports the following new **Synchronization** functions:

- Lossy Enqueue Element
- Wait on Notification from Multiple and Notifier History
- Wait on Notification with Notifier History

Microprocessor SDK Documentation

The Microprocessor SDK includes the following documentation in addition to this document:

- The *LabVIEW Microprocessor SDK Porting Guide*, available by selecting **Start»All Programs»National Instruments» LabVIEW 8.6»LabVIEW Manuals** and opening `MicroprocessorSDK_Porting_Guide.pdf`, contains the information you need to port LabVIEW to a new target.
- The *LabVIEW Embedded Development Module Target Distribution Guide*, available by selecting **Start»All Programs»National Instruments»LabVIEW 8.6»LabVIEW Manuals** and opening `EMB_Distribution_Guide.pdf`, contains the information you need as an OEM or VAR if you want to bundle and resell LabVIEW and your target.
- The readme file, available by selecting **Start»All Programs» National Instruments»LabVIEW 8.6»Readme** and opening `readme_MicroprocessorSDK.html`, contains known issues.
- The *LabVIEW Help*, available by selecting **Help»Search the LabVIEW Help**, contains reference information about LabVIEW palettes, menus, tools, VIs, and functions. The *LabVIEW Help* also

contains step-by-step instructions for using LabVIEW. The *LabVIEW Help* uses (Microprocessor SDK) in the index to indicate topics specific to the Microprocessor SDK. The *LabVIEW Help* uses (Embedded Targets) in the index to indicate topics that are relevant to all embedded targets.

Where to Go for Support

In addition to the Microprocessor SDK documentation, you also have access to the LabVIEW Embedded Discussion Forum and training.

- The LabVIEW Embedded Discussion Forum provides a discussion forum for the Microprocessor SDK and other LabVIEW Embedded products. You can access the forum by visiting `ni.com/info` and typing `edmforum`.
- On site training at National Instruments corporate headquarters or Web-based training is available upon request.

National Instruments corporate headquarters is located at 11500 North Mopac Expressway, Austin, Texas, 78759-3504. National Instruments also has offices located around the world to help address your support needs. For telephone support in the United States, create your service request at `ni.com/support` and follow the calling instructions or dial 512 795 8248. For telephone support outside the United States, contact your local branch office:

Australia 1800 300 800, Austria 43 662 457990-0,
Belgium 32 (0) 2 757 0020, Brazil 55 11 3262 3599,
Canada 800 433 3488, China 86 21 5050 9800,
Czech Republic 420 224 235 774, Denmark 45 45 76 26 00,
Finland 358 (0) 9 725 72511, France 01 57 66 24 24,
Germany 49 89 7413130, India 91 80 41190000, Israel 972 3 6393737,
Italy 39 02 41309277, Japan 0120-527196, Korea 82 02 3451 3400,
Lebanon 961 (0) 1 33 28 28, Malaysia 1800 887710,
Mexico 01 800 010 0793, Netherlands 31 (0) 348 433 466,
New Zealand 0800 553 322, Norway 47 (0) 66 90 76 60,
Poland 48 22 3390150, Portugal 351 210 311 210, Russia 7 495 783 6851,
Singapore 1800 226 5886, Slovenia 386 3 425 42 00,
South Africa 27 0 11 805 8197, Spain 34 91 640 0085,
Sweden 46 (0) 8 587 895 00, Switzerland 41 56 2005151,
Taiwan 886 02 2377 2222, Thailand 662 278 6777,
Turkey 90 212 279 3031, United Kingdom 44 (0) 1635 523545

Printed in Ireland. 324064D-01 Jun08